Programmation Python

De la base aux concepts avancés de Python

Audrey Kouessi FANGNON & Johannès HOUNSINOU

Bénin Excellence - Fondation Vallet / FSS COTONOU

19 octobre 2024

Plan du cours

1. Chapitre 2 : Les fondamentaux de la programmation Python
   - L'arithmétique et les opérateurs (relationnel et logique)
   - Les chaines de caractères
   - Les variables et les types

L'arithmétique.

Opération arithmétique

Python permet de faire les opérations arithmétiques sur des valeurs entières (int) ou fractionnaire (float ou flottant)

+	Addition
-	Soustraction
*	Multiplication
/	Division décimale
//	Division entière
**	Exponentiation ou Puissance
%	Modulo

Table 1: Les opérateurs arithmétiques

Coding Time - Opération arithmétique

Testons les opérations ci-dessous après avoir deviné les résultats et amusons nous à comprendre les opérateurs.

```
>>> 25 + 85
>>> 1.5 - 4
>>> 3.14 * 3
>>> 9 / 2
>>> 9 // 2
>>> 9 % 2
>>> 3 ** 4
```

L'arithmétique.

Associativité et priorité

La plupart des opérateurs sont associatifs à gauche sauf l'exponentiation.

**	Associatif à droite
* ; / ; // et %	Associatif à gauche
+ et -	Associatif à gauche

Table 2: Associativité et précédence des opérateurs depuis le plus prioritaire vers le moins prioritaire.

Parenthèse

Les parenthèses sont utilisées pour rendre lucide les opérations ou pour modifier l'ordre d'évaluation.

Coding Time - Associativité et priorité

Testons l'associativité et la priorité des opérateurs

```
>>> 2 - 3 - 4
>>> (2 - 3) - 4
>>> 2 - (3 - 4)
>>> 2 ** 3 ** 2
>>> (2 ** 3) ** 2
>>> 2 ** (3 ** 2)
>>> 2 + 5 * 3
>>> 2 * 5 ** 2
>>> -2 + 12 % 5
```

Les opérateurs relationnels.

Les opérateurs de comparaison, appelés opérateurs relationnels en Python compare des valeurs et renvoie les booléen Vrai (True) ou Faux (False).

a < b	la valeur de a est-elle strictement inférieure à la valeur de b
a > b	la valeur de a est-elle strictement supérieure à la valeur de b
a >= b	la valeur de a est-elle supérieure ou égale à la valeur de b
a <= b	la valeur de a est-elle inférieure ou égale à la valeur de b
a == b	la valeur de a est-elle égale à la valeur de b
a != b	la valeur de a est-elle différente de la valeur de b

Table 3: Les opérateurs relationnels

Coding Time - Les opérateurs relationnels.

Testons les opérateurs relationnels.

```
>>> 2 < 5
>>> 10 >= 10
>>> 10 > 8
>>> 7 != 7
>>> 7 == 7
```

Les opérateurs logiques.

and-or-not

La table de vérité des opérateurs and (qui correspond au « et » en français) or (« ou » en français) et not (« non » en français) définit la sémantique de chacun des trois opérateurs logiques.

a	b	not a	a and b	a or b
False	False	True	False	False
False	True	True	False	True
True	False	False	False	True
True	True	False	True	True

Table 4: Table de vérité de : and - or - not

Les opérateurs logiques.

Loi de **De Morgan**

- not (a or b) est équivalente à (not a) and (not b)
- not (a and b) est équivalente à (not a) or (not b)

Coding Time - Les opérateurs.

Testons les expressions suivantes.

```
>>> not False and True
>>> not (False and True)
>>> True and False or True
>>> True and False or True
>>> 4 - 2 <= 5 + 8
>>> 5 >= 10 and 5 != 8
>>> 2 == 2 and not 3 != 5
```

Les chaines de caractères

string

- Les chaines de caractère appelé ***string*** en anglais permettent à ***Python*** de traiter du texte.
- Pour écrire un ***string***, il suffit d'entourer le texte saisie de ***doubles*** ou ***simples quotes*** *(apostrophes)*.
- ***Python*** autorise l'écriture du texte sur plusieurs lignes en entourant le texte saisie par des ***triples quotes*** : ''' ou bien """.

Exemples :

```
>>> 'Benin Excellence'
>>> "Python est super-puissant"
>>> """Python est le langage
... informatique que je préfère"""
```

Les chaines de caractères

Concaténation

La ***concaténation*** (+) est l'opération qui consiste à *coller deux textes ensemble* pour n'en former qu'un.

Répétition

La concaténation du même texte plusieurs fois est possible grâce à l'opérateur (*).

Caractère d'échappement

Le caractère d'échappement en python est : l'antislash (barre oblique inverse \) et permet d'afficher le caractère apostrophe et de faire certain ajustement.

Coding Time - Les chaines de caractères.

Testons les expressions suivantes.

```
>>> 'Je suis fière d'apprendre'
>>> 'je suis fière d\'apprendre'
>>> "Je suis fière d'apprendre"
>>> 'Bénin' + ' ' + 'Excellence'
>>> 'Bénin Excellence ' * 5
>>> ''' Je suis un
... un génie en informatique '''
```

Les variables et les types

Les variables

Une variable est un nom servant à repérer l'emplacement de la mémoire où le programme stocke une donnée qu'il manipule.

Déclaration et initialisation

La déclaration et l'initialisation d'une variable se font simultanément. nom_variable = valeur

Après l'***assignation***, (par exemple x = 47), la variable assignée a une valeur et un type (ici x vaut la valeur 47 et son type est int [entier])

Les variables et les types

Types des variables

On distingue plusieurs types de variables en Python dont les quatre principaux sont :

- int : variable recevant une valeur numérique entière.
- float : variable recevant une valeur numérique réelle.
- str : variable recevant un caractère.
- bool : variable recevant un booléen.

Le type d'une variable peut être connu grâce à la fonction type()

```
>>> nombre = 29
>>> type(nombre)
>>> valeur_logique = True
>>> type(valeur_logique)
```

Les variables et les types

Attention

La variable qui reçoit la valeur d'une ***assignation*** est à gauche du symbole = (symbole d'affectation)

Règle de nommage

- Le nom peut être composé de lettres (majuscules ou minuscules), de chiffres et du symbole underscore « _ » mais ne peut pas commencer par un chiffre.
- Python est sensible à la casse.
- Certains noms sont des ***mots réservés*** (mots-clés) Python. Ainsi ***continue = 0*** ne fonctionne pas car ***continue*** est un ***mot-clé Python***.

Les variables et les types

Mots-clés

and	del	from	none	True
as	elif	global	nonlocal	try
assert	else	if	not	while
break	except	import	or	with
class	False	in	pass	yield
continue	finally	is	raise	
def	for	lambda	return	

Table 5: Mots-clés : Python3

Les variables et les types

Affichage

La fonction ***print()*** est utilisée pour afficher du texte, des valeurs, des variables ou tout autre contenu à la console.

Saisie

La fonction ***input()*** affiche un message à l'utilisateur et attend que l'utilisateur entre des données puis retourne les données entrées sous forme de ***string***.

Exemples :

```
>>> age = input("Entrer votre age : ")
>>> type(age)
>>> age = int(age)
>>> print("Votre age est : ", age)
```

Coding Time - Saisie et affichage

Sachant que la fonction float() permet de convertir un string en flottant, complète le code suivant:

```
>>> taille = _ _ _ input("Entrer votre taille : ")
>>> _ _ _ ("Votre taille est : ", _ _ _ )
```

Les variables et les types

Modification et transtypage d'une variable

L'instruction d'affectation (=) réalise la création et la mémorisation de la variable. La valeur mémorisée dans une variable peut être modifiée en faisant semblant de créer une nouvelle variable du même nom.

Exemples :

>>> x = 4 crée une variable de nom x, de type int et valant 4

>>> x = 78 modifie la valeur de la variable x qui et désormais 78

>>> x = 5.32 modifie le type de la variable x qui est désormais float et sa valeur qui passe à 5,32

Les variables et les types

Copie de variable

Il est possible de copier la valeur d'une variable dans une autre variable par une simple instruction : variable_2 = variable_1

Exemples :

```
>>> x = 22
>>> y = x
>>> print(y)
```

Les variables et les types

Opérations sur les variables

Tous les opérateurs vu précedemment sont valable avec les variables.

Exemples :

```
>>> x = 10
>>> y = 6
>>> x + y
>>> x * y
>>> x % y
>>> x \ y
```